GCCS System Integration Support

RDA Maintenance Manual
(Update)

September 27, 1996

Prepared for:

DISA/D611
ATTN: Ms. Claire Burchell
45335 Vintage Park Plaza
Sterling, VA 20166-6701

Contract Number: DCA 100-94-D-0014
Delivery Order Number: 204, Task 5
CDRL Number: A071

Prepared by:

Computer Sciences Corporation
Defense Enterprise Integration Services
Four Skyline Place
5113 Leesburg Pike, Suite 700
Falls Church, VA 22041

THIS DOCUMENT IS UNCLASSIFIED

TABLE OF CONTENTS

RDA MAINTENANCE MANUAL
(UPDATE)

TABLE OF CONTENTS (CONTINUED)

APPENDICES

1.0 SCOPE

1.1 Identification

The Requirements Development and Analysis (RDA) System, Software Programmer's Manual (SPM): Maintenance Documentation, was developed in accordance with Section 2.0, Reference item a. RDA is a subsystem of the Joint Operation Planning and Execution System (JOPES) component of the Global Command and Control System (GCCS), Version 2.1.

1.2 Application Overview

RDA provides a capability to create, add, modify, delete, and generate output on deployment-related information contained in an Operation Plan (OPLAN) Time-Phased Force and Deployment Data (TPFDD). See Section 2.0, Reference items b., c., and d., for current and legacy system data descriptions.

RDA is the result of migrating and integrating applicable portions of the Joint Operation Planning System (JOPS) and the Joint Deployment System (JDS). This TPFDD edit capability is a critical tool for deliberate or peacetime planning and time-sensitive or Crisis Action System (CAS) planning. From a functional viewpoint, the legacy systems (JOPS/JDS) procedures for TPFDD development and edit were satisfactory; however, technical issues associated with the GCCS development necessitated a migration to a modern, state-of-the-art operating environment.

The specific types of functionality include:

a. Plan Population and Maintenance,
b. Requirements Generation and Maintenance,
c. Availability of Unit Information,
d. Force Module (FM) Development and Maintenance,
e. Availability of Reference File Information, and
f. Pre-Defined Reports/Retrieval Generation.

The need for a replacement of the legacy systems has been long recognized. In the summer of 1994, a migration strategy was developed (see Section 2.0, Reference items e., and f.), which addressed the various legacy elements that required migration and provided an overview of the time and effort required to migrate to the GCCS that was (and is) under development. After several sessions between the overall Government sponsors and end users, a TPFDD editor replacement project (later named RDA) was initiated. The project is directed by the Defense Information Systems Agency (DISA) with the user community being represented by the JOPES User Review Panel (URP) (see Section 2.0, References, item g.).

1.3 Document Overview

This document details the information required by MIL-STD-498 (Section 2.0, Reference item a.), Data Item Description (DID), DI-IPSC-81441. The main document contains the Scope, References, Notes and forward references to the remaining information in the appendices. The appendices contain the detailed technical information to support RDA maintenance. The majority of this information is accessible on-line. Appendix A contains the description of Work Packages. Appendix B describes the Functional Units of the RDA. Appendix C provides the Requirements Traceability (unchanged since last submission). Appendix D lists the RDA Objects Calling Other Objects (unchanged since last submission). Appendix E shows the RDA Objects Called by Other Objects. Appendix F describes the Hardware Utilization, and Appendix G lists all the RDA Software, Executables, and Segmentation scripts.

This document is unclassified and there are no security considerations associated with its use.

2.0 REFERENCES

The following documents are referenced in this SPS:

a. Department of Defense (DoD), Military Standard-498 (MIL-STD-498) Software Development and Documentation, Software Product Specification (SPS), Data Item Description (DID) DI-IPSC-81441, December 5, 1995.

b. Joint Staff, Joint Operation Planning and Execution System (JOPES) Development and Integration Maintenance Manual: Scheduling and Movement (S&M) GCCS Core Database Maintenance Manual, August 25, 1994.

c. Joint Staff, Joint Operation Planning System (JOPS) Time-Phased Force and Deployment Data (TPFDD) and Related Files, (SPM DS 143-87), April 1, 1987.

d. Joint Staff, Joint Deployment System Data Base Specification, (TD 18-17), September 30, 1988.

e. Joint Staff, Joint Operation Planning and Execution System (JOPES) Migration Strategy (Draft), Version 1.0, August 8, 1994.

f. Defense Information Systems Agency, Migration Engineering Strategy Guide Near-Term Migration Strategy, August 31, 1994.

g. Defense Information Systems Agency, Worldwide Military Command and Control System (WWMCCS) Intercomputer Network (WIN) Teleconference GCCS NEWS Message, Number 390, Subj: 21-23 Feb Meeting of JOPES URP, March 1, 1995.

h. Joint Pub 1-03.21 - to become CJCSI 3150.16, Joint Operation Planning and Execution System Reporting Structure (JOPESREP), May 24, 1994.

I. Defense Information Systems Agency, JOPES Users Guide, May 16, 1996.

j. Defense Information Systems Agency, Global Command and Control System (GCCS) Integration Standard, Version 1.0, Washington D. C., October 26, 1994.

k. Computer Sciences Corporation, Systems Engineering Division, GCCS JOPES Migration Engineering and Implementation, Software Development Plan (SDP), Falls Church, Virginia, June 15, 1995.

l. Computer Sciences Corporation, Systems Engineering Division, JOPES Software Development Standards, Falls Church, Virginia, July 25, 1995.

m. Defense Information Systems Agency, Defense Systems Support Organization, Joint Operation Planning and Execution System (JOPES) Reference File Paging (RFP) Users Manual, Computer System Manual, CSM UM 297-92, September 28, 1992.

3.0 REQUIREMENTS

The RDA application requirements are fully documented in Appendix C, Requirements Traceability. This application operates within the context of the GCCS Desktop as a standard executable. The JOPES High Level System Navigation (JNAV) process is capable of launching the processes for the RDA application.

RDA is a versatile, user friendly, requirements development and analysis modeling capability that supports military operation planners in developing or modifying the force and nonunit requirements (cargo and personnel) for deliberate planning or for crisis deployment operations. To accomplish this, RDA uses an integrated set of automated tools and the JOPES Core database supporting joint operators and planners in both peacetime (deliberate) and time-sensitive (crisis action) planning conditions. It affords the user an opportunity to quickly develop and analyze proposed Courses of Action (COAs) in relation to asset allocations and TPFDD modifications.

RDA operates directly off the JOPES Core database; thus, it is connected to all systems using this database. Presently, there are other systems within GCCS sharing this relational database in the client-server environment. These systems are the Joint Flow and Analysis System for Transportation (JFAST), Logistics Sustainment Analysis and Feasibility Estimator (LOGSAFE), RDA, Scheduling and Movement (S&M), Pre-Defined Reports (PDR), Transportation Component Command (TCC) External System Interfaces (ESI), Ad Hoc Query (AHQ), Joint Engineer Planning and Execution System (JEPES), Medical Planning and Execution System (MEPES), GCCS Status of Resources and Training System (GSORTS), Information Resource Manager (IRM), and Force Augmentation Planning and Execution System (FAPES).

Through the sharing of functional applications via this core relational database across different computer system platforms, RDA gives joint operators and planners the ability to edit and analyze TPFDDs for accuracy and transportation feasibility. It supports rapid manipulation of the TPFDD and graphically displays and compares data for rapid analysis. Because of the advantages available via the use of a relational database and the client-server architecture, the requirement for external interfaces is met through the JOPES Core database.

RDA updates to site's JOPES Core database is accomplished by one of the following methods:

a. Update functions to all database elements for a specific OPLAN, except date range changes, are accomplished by directly updating the sites' local database and generating appropriate update transactions. On successful completion of the local database update, transactions are committed to the Transaction Processing (TP) Send Queue.

b. Updating database date range changes is accomplished by generating the appropriate update transactions. The External Transaction Processor (XTP) updates the local database, and on successful completion of the local database update, transactions are committed to the TP Send Queue.

The Transaction Distribution System (TDS) extracts transactions from the Send Queue and distributes transactions across the network for database update synchronization to all alternate sites maintaining the affected OPLAN. In certain instances, a large volume of transactions may be generated for specific update functions encompassing an entire OPLAN, e.g., Update Plan Identification Number (PID) from Type Unit

Characteristics File (TUCHA). These types of mass updates may cause a slight delay in database transaction update synchronization.

The system follows the constraints of previous, legacy system software and Joint Operation Planning and Execution System Reporting Structure (JOPESREP) (see Section 2.0, item h.) with the exceptions described. These exceptions have been directed and coordinated with the JOPES URP and are listed below. Information within the parentheses in each entry below indicates the specific reference to document(s) listed in Section 2.0.

The independent force category for a Unit Line Number (ULN) has been expanded to permit two character values (see Section 2.0, item h. Table I-3).

A fatal error will not result if the split personnel record includes cargo for Split shipments. Similarly, the split cargo record will not cause a fatal error if personnel are include, (see Section 2.0, item h., Table I-3; Table I-28/ Nos. 39, 46; Appendix B, Functional Units).

Intermediate stop location codes have been updated to the new database values (see Section 2.0, item h., Table I-13).

PID, Force Requirement Number (FRN), Cargo Increment Number (CIN), and Personnel Increment Number (PIN) Reserved assignments are not enforced or edited (see Section 2.0, item h., Table I-27).

FRN hierarchy are not enforced or edited (see Section 2.0, item h., Table I-28).

No edits are performed to enforce no force movement characteristics, no force requirement and routing data, or indicated nonunit data for example records with destination preferred mode of "Z" (see Section 2.0, item h., Appendices B and C).

No edits are performed on Alternate Ports & Ports of Support (see Section 2.0, Reference item h., Appendix B, Functional Units).

3.1 Executable Software

Refer to Appendix G, Executable Software, Source Files, Segmentation Scripts, Paragraph G.1.

3.2 Source Files

Refer to Appendix G, Executable Software, Source Files, Segmentation Scripts, Paragraph G.2.

3.3 Packaging Requirements

This paragraph has been tailored out.

3.4 Segmentation Scripts

Refer to Appendix G, Executable Software, Source Files, Segmentation Scripts, Paragraph G.3.

4.0 QUALIFICATION PROVISIONS

The RDA application is delivered on magnetic tape media, using the standard UNIX "tar" format. The current version identification can be verified through manual examination of the version files /h/RDA/SegDescrip/VERSION and /h/RDASRV/SegDescrip/VERSION.

5.0 SOFTWARE SUPPORT INFORMATION

5.1 "As Built" Software Design

5.1.1 Design Decisions

RDA is the reengineering of a prototype (Dynamic Analysis and Replanning Tool (DART)) planning tool coupled with the migration of business rules existing on the legacy WWMCCS mainframe system. The target platform is a client/server environment using an ORACLE RDBMS and the MOTIF X-Windows system. The RDA System was required to provide rapid user feedback for changes to the database and to support the distribution of transactions representing the changes made by the user. Additionally, support was required for multiple users in both a local and a distributed environment.

The RDA design utilizes a two-step update approach where database updates are made directly to the database and JDS formatted transactions representing the updates are delivered to the Transaction Distribution System (TDS). If either step fails, the entire process is rolled back and no update is performed.

To support the current set of logical business rules and prepare for future rule definitions, the logical error checking portion of the system was encapsulated in a series of stored procedures in the database. This is found in the Verification Engine work package.

5.1.2 Architecture

The RDA architecture can be described in a hierarchical fashion. Each element in the hierarchy consists of a set of physical files. The highest elements are termed work packages. Work packages themselves consist of Logical Units (LUs) which are composed of Functional Units (FUs).

5.1.2.1 Work Packages

The RDA work packages are RDACargo, RDADevTools, RDAForceModules, RDAMain, RDA Mapping, RDAMergeCompare, RDAReferenceFiles, RDARequirements, RDASelect, RDATimeline, RDAUtility, Transaction Generation, and Verification Engine as shown in Figure 5.1.2.1-1, RDA Work Packages. Each work package with its association LU and list of FUs is described in Appendix A, Work Packages.

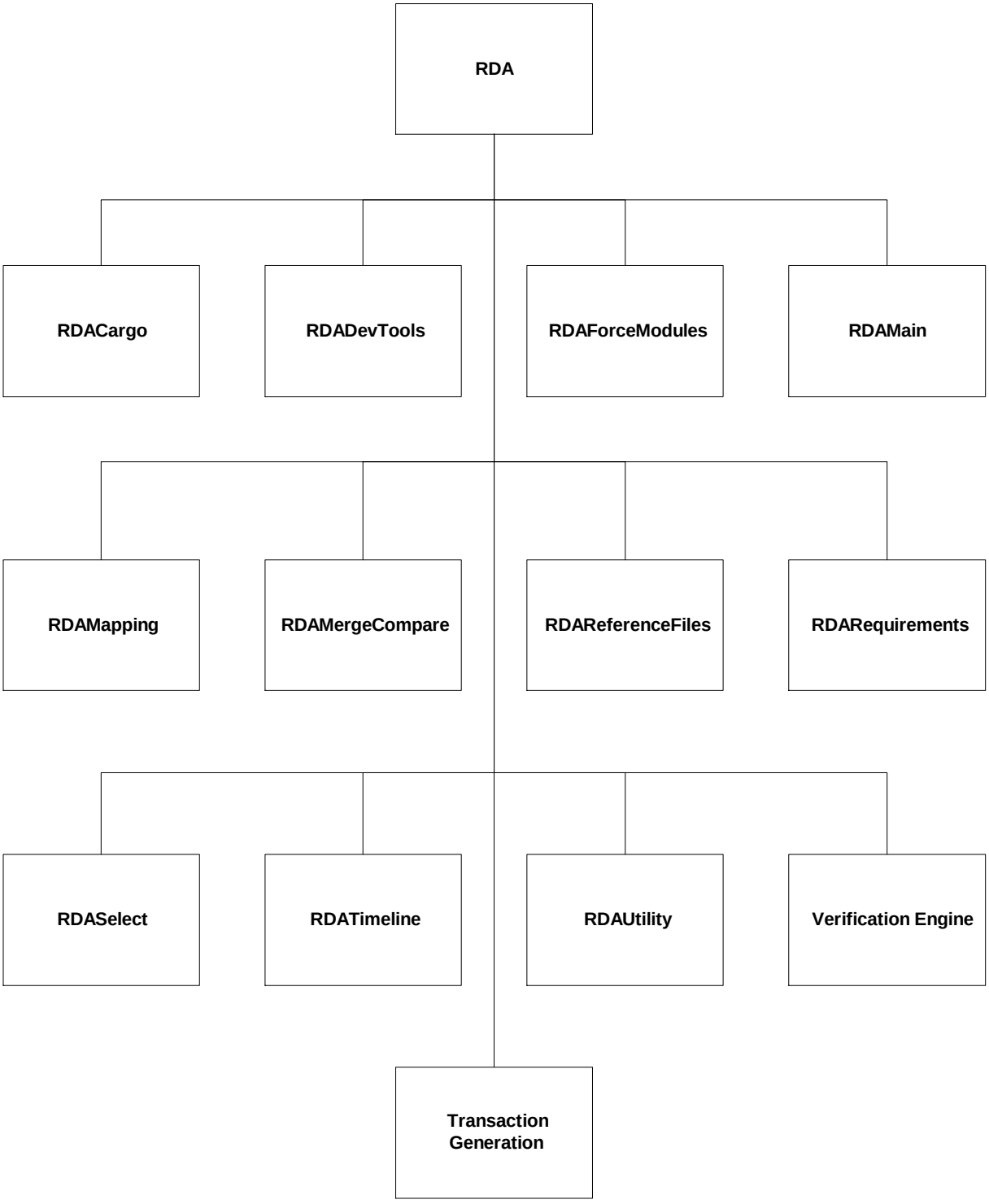

Figure 5.1.2.1-1. RDA Work Packages

5.1.2.2 Functional Units

See Appendix B, Functional Units.

5.1.3 Environmental Variables

RDA uses the following environment variables:

GAINHOME	Defines the directory path where the RDA System files are located.
DBVENDOR	Defines the database vendor supplying the Relational Database Management System (RDBMS) software.
DBSERVER	Defines the location of the hardware platform on which the database server is executing. The value of this environment variable names a section in the $GAINHOME/lib/sqlhosts.txt file.
DBUSER	Defines the user id used for logging into the database. When using ORACLE OPS$ accounts, this value must be '/'.
DBPASSWORD	Defines the database login password for the user named in $DBUSER. When using ORACLE OPS$ accounts, this value must be empty (').
DBTIMEOUT	Defines the time in seconds to wait for any SQL command to return.
RDAHELPENGINE	Defines the path to the help engine used for displaying help windows.
RDAHELPDATA	Defines the path to the help files.
RDA_HOME	Location of the RDA segment.
RDA_PRINT_P1	Defines the first part of the command used to print RDA output files that are not part of PDR.
RDA_PRINT_P2	Defines the second part of the command used to print RDA output files that are not part of PDR.
PDR_HOME	Location of the PDR segment.
TWO_TASK	ORACLE environment variable pointing to the database server.
DBENTRY	Defines sqlhosts.txt entries in the temporary sqlhosts.txt file that is created for use with ORACLE OPS$ accounts.
DBSERVERNAME	Defines the "SERVER=" lines in the temporary sqlhosts.txt file that is created for use with ORACLE OPS$ accounts.

DBENVLIST	Defines environment variables necessary for the temporary sqlhosts.txt file that is created for use with ORACLE OPS$ accounts.
DBPROXY	Defines the ORACLE proxy executable in the temporary sqlhosts.txt file that is created for use with ORACLE OPS$ accounts.
VTXDIR	Defines the vortex library location in the temporary sqlhosts.txt file that is created for use with ORACLE OPS$ accounts.
DBSTARTPORT	The port number that will be used as a start point for searching for an available port to use for starting the gmnetd.exe executable.
DBPORTS	The number of ports (from the DBSTARTPORT value) that will be searched.
RDA_JSIT_HOME	Location of JSIT scripts
WISH	Location of windowing system for JSIT

If no port between DBSTARTPORT and DBSTARTPORT+DBPORTS is found, RDA will not start and an error message will be generated. The case (upper vs. lower) of the environment variables is important.

Printing within RDA is implemented using the two RDA_PRINT_P1 and RDA_PRINT_P2 environment variables. Defaults are provided and should be modified to suit the individual site.

These environmental variables are set in /h/RDA/RDA.env which is sourced by the RDA launch program /h/RDA/progs/RDA_run.

5.1.4 Database Variables

RDA creates several ORACLE database objects including users, roles, and packages. Files that create or control these objects are described below.

5.1.4.1 RDA User and Role

a. create_rda_role.sql (/h/RDASRV/sql),
b. drop_rdauser.sql (/h/RDASRV/sql),
c. grant_rda_role.sql (/h/RDASRV/sql),
d. gsorts_rda_role.sql (/h/RDASRV/sql), and
e. rda_role.sql (/h/RDASRV/sql).

5.1.4.2 RDA Database Packages

See Appendix G, Executable Software, Source Files, Segmentation Scripts, Paragraph G.2.2, ORACLE PL/SQL Packages, for a list of database packages.

5.1.5 External System Interfaces

The External Transaction Processor (XTP) is called from the Shift TPFDD Dates function to update the database and process the transactions generated by the function. The transactions processed this way are Force Module Requirement Transaction (JJDSDT), Requirement Date Transaction (DATEBT), and Force Module Delete ID Transaction (DLFMDT). The parameters required for running XTP are:

a. an error directory in which the error logs will be created. - In RDA, this will be the user's home directory;

b. a message log filename in which messages are generated for invalid and valid transactions as they are processed - this will be shiftDate.log;

c. a transaction filename containing all the transaction records to be processed - this will be shiftDate.tran; and

d. a report filename containing the final xtp report - this is shiftDate.rpt.

5.2 Compilation/Build Procedures

To create an RDA distribution, two segment structures must be produced (RDA [client], RDASRV [database server]). The procedures for generating these two segments are presented below.

A special user id has been reserved for the creation of RDA segments. This user ID is "rdart" (RDA Runtime). The rdart user ID has its own set of RDA libraries that are controlled and managed solely by the rdart user. No development work should be performed in this separate Gain environment.

5.2.1 Determine Build Location for Segments

A full RDA and RDASRV segment pair occupy approximately 90 MB of disk space. In addition, the runtime generation process consumes an additional 40 to 50 MB during the build process. This additional space is used for temporary files and so is reclaimed after the runtime generation is complete.

Use the following procedure to determine if enough disk space is available in the default build area (/home/rdart/distrib):

Ensure that there is enough disk space for the segments by executing the following command.

df -k | grep rdart

If the above command shows that usage on the rdart partition is above 89%, another destination must be found for the runtime. Suggestions are /temp and /scratch. Check each of these to see which is the best choice.

5.2.2 RDA Client Segment Build Procedure

Generating an RDA client segment is composed of the following steps:

a. Copy developer files into "rdart" holding directories,
b. Import files into rdart Gain environment,
c. Build Gain Momentum runtime,
d. Update segmentation specific files, and
e. Create and populate supporting segmentation directories and files.

Each step is described in greater detail in the paragraphs below.

5.2.2.1 Copy Developer Files into "rdart" Holding Directories

As developers complete work on Gain applications, they are required to export the completed application files to Gain interchange files (.if). Following the export, the developer sends a message to the "rdart" user specifying the PR/SCR number completed and the associated files and their locations. These files are used to move the work from the development environment to the runtime creation environment managed by the "rdart" user.

A special holding area is maintained within the "rdart" user directory for storing these .if files before they are imported to the Gain runtime environment. This area is under the `~/src` directory and contains a directory for each developer (by developer user id).

The following steps are used to move the files to the holding area:

Steps

a. Move to the `src` directory under the "rdart" home directory.

b. Remove any pre-existing source files from the last runtime generation using the following C-Shell Script:

```
source rmoldsrc
```

The above script will remove all files from the developer directories under the `src` directory. This step should only be executed once prior to the building of each runtime. This list is coded into the rmoldsrc script, and must be modified when the developer team changes.

c. Print the mail messages that contain the developer notification messages. Go through each message and highlight each file that is to be copied.

d. Copy each file from the developers' export directory to the corresponding holding directory using the `cpp` alias defined for the "rdart" user. This alias preserves file date information which can be used to cross check files between the developer's directory and the holding directory:

    cpp ~<user ID>/rtexport/* <user ID>

    Example (assuming current directory is the ~/src directory):

    cpp ~ray/rtexport/* ray

    Double check the contents of the holding directory to ensure that the contents of the directory match the mail message contents. If a discrepancy occurs, check file dates and remove extraneous files. Copy all the appropriate export files for all the mail messages.

e. Create a log of all the files that are ready for import, and deconflict the log to reconcile duplicates that may have been submitted by two different developers. Use the following command to generate a file log:

    ```
    cd ~rdart/src
    ls -l -R > filesxx.txt
    ```

    Print the filesxx.txt file and examine each of the developers directories to see if the same application has been submitted more than once. If the same application appears multiple times, examine the top of each file to see which one was exported last. When Gain creates an export file, it inserts a datestamp in the beginning to indicate when the export was created. Strike through all but the latest application export.

f. Use the resulting marked up list as the list of import files for the next step.

5.2.2.2 Import Files into "rdart" Gain Environment

The "rdart" Gain environment is a separate installation and library structure that is used to maintain the production system. This environment mirrors the development environment with identical library names and organizations. Changes to the libraries are executed through the import of modification application export files originating from the development environment.

This procedure assumes that the import files have been identified through the procedure defined in Paragraph 5.2.2.1.

Import the files using the following steps:

Steps:

a. Logon to the computer on which the RDA Gain Momentum files are available and start Gain Momentum by entering the following command:

    rtgmOps31 &

b. Open the library that contains the application that will be replaced.

c. Delete the application from the library.

d. Import the replacement using the File | Import option in the library window.

e. Check in the newly-imported application. When importing an application, Gain does not also check-in the application. The check-in must be performed manually.

f. Repeat steps 2 through 5 for each of the files to be imported.

g. As a safety measure, the entire contents of the RDAUtility and RDADevTools libraries are exported from the development environment and imported to the runtime environment. This is done to ensure that changes to all support tools are captured. To do this, export the entire library to a .if file. When importing a library, Gain will create a folder with the same name as the library in the library. This folder contains the contents of the exported library.

    Before the new applications can be used, they must be moved out of the new folder and up to the main level of the library (replacing the existing applications). To do this, first delete all of the original applications from the library, leaving only the newly imported folder. Open the folder by double-clicking. Drag select the contents of the folder. All applications should be high-lighted in reverse video. Click and hold on the high-lighted applications and drag to the library window (containing the folder). Release and the selected applications will move from the folder to the library. The folder should be empty, and can be deleted.

h. Proceed to build the Gain Momentum runtime.

5.2.2.3 Build Gain Momentum Runtime

The Gain momentum runtime generation process is fairly straight forward. Use the following steps to generate a runtime.

Steps:

a. Logon to the computer on which the Gain Momentum files are available as "rdart" by entering the following command:

su - rdart

b. Enter the password.

c. Open the "Make Runtime" folder. This folder is the start point for the RDA runtime.

d. Deselect any applications that have been selected by clicking the mouse in an empty portion of the "Make Runtime" folder window, then begin building the runtime by clicking **{File}** and then clicking **{Make Runtime}**.

e. In the "Included Tools" section of the Make Runtime dialog box, scroll to the "StartRDATool" and CTRL+click the left mouse button to add it to the runtime. Also add "RDAProfiler".

f. In the "Contents" list box of the "Startup Actions" section, highlight "application RDASkeleton". Click **{Start With}** next to the "Startup Action" label underneath the list box. In the text entry box next to **{Start With}**, press the arrow button and select "StartRDATool."

g. Deselect "End With Last Data Manager."

h. Click **{File System Settings}**.

i. In the directory entry at the top of the form, enter the directory that will be used for the runtime. All segment deliveries to CM are under a master directory that uses the pattern **rt**yymmdd. For example, if the date on which the runtime is generated is September 28, 1995, the master directory will be rt950928.

The directory entry at the top of the form should point to the fully qualified directory name determined above. Under this directory will be two subdirectories that contain the two segments for RDA. The RDA subdirectory will contain the client segment and the RDASRV subdirectory will contain the server segment.

j. For the target directory, enter the following: RDA.

This will be the top level of the RDA client segment directory structure. During runtime generation, Gain will create the RDA directory. If the directory already exists, Gain will delete the entire contents of the directory before proceeding. You will be prompted before the deletion occurs.

k. Click **{Scan Libraries}** in the "Libraries To Include" section.

l. When the scan completes, select the following libraries by highlighting them with the mouse while holding down the **[Ctrl]** key:

PredefinedReports,
and all libraries that begin with "RDA".

m. In the "Platforms" section, hold down the CTRL key and click the left mouse button on hp700. Both hp700 and sol2s should be selected. This builds both a Solaris and an Hewlett-Packard (HP) runtime.

n. Click **{Non-Sybase RDBMS Access Settings}** and verify that both of the following lines are highlighted:

```
oracle7  hp700  var/vortex/bin/hp700/gmoracl7.exe
oracle7  sol2s  var/vortex/bin/sol2s/gmoracl7.exe
```

o. Click **{Make Runtime}.**

p. A dialog box containing space requirements information appears. Click **{Continue}**.

q. Click **{OK}** to start the runtime build. At this point, you have to monitor the build process to answer a few dialogs concerning the lack of a prototype definition and a request to determine if this is the last version before saving. Answer yes or confirm all dialogs.

r. After 30-45 minutes, a dialog box appears, saying that the build was successful. Click **{OK}** in response.

s. Click **{Dismiss}** to exit the "Make Runtime" screen.

t. Export all the libraries using the "MiscOps" tool located in the rdart library. To use the tool, populate the center list box with the library names, and click the "Compress Exports" check box. Click the **{Export Libs}** button. This will deposit .Z files in the "rdart" home directory.

u. Exit Gain Momentum by clicking **{File}** and then clicking **{Quit Gain Momentum}**.

v. Proceed to the next step to add the segmentation files to the runtime structure.

5.2.2.4 Update Segmentation Specific Files

The segment files /home/rdart/devsrc2/RDA/SegDescrip/VERSION and /home/rdart/devsrc2/RDA/SegDescrip/ReleaseNotes must be updated with the correct information for the release.

a. From UNIX prompt: su - rdart

b. Enter password.

c. codemgrtool&

d. Highlight work area (e.g., /home/diana/devsrc2/RDA).

e. Using right mouse button choose Transactions/Bringover/Update.

f. With right mouse button choose Edit/Select All.

g. With left mouse button click on SegDescrip/ReleaseNotes and SegDescrip/VERSION.

h. With right mouse button choose Edit/Delete.

i. With left mouse button press Bringover. When complete close status window and Bringover window.

j. From main codemgrtool window double click on work area (e.g., /home/diana/devsrc2/RDA) and then double click on SegDescrip.

k. Using both mouse buttons click on ReleaseNotes and VERSION.

l. With right mouse button choose Commands/Check Out. When complete close down codemgrtool completely.

m. Using UNIX editor of choice, edit ReleaseNotes and VERSION files with appropriate changes for the release. This information includes items such as release, version, build date, and delivery date.

n. From UNIX prompt: codemgrtool&

o. Using left mouse button double click on work area and then double click on SegDescrip.

p. Using both mouse buttons click on ReleaseNotes and VERSION.

q. With right mouse button choose Commands/Check In.

r. Using right mouse button choose Transactions/Bringover/Putback.

s. With right mouse button choose Edit/Select All.

t. With left mouse button click on SegDescrip/ReleaseNotes and SegDescrip/VERSION.

u. With right mouse button choose Edit/Delete.

v. With left mouse button press Putback. When complete close status window and Putback wndow.

5.2.2.5 Create and Populate Supporting Segmentation Directory

After all developer changes to non-Gain client files have been completed, checked in and putback into codemgrtool, the final phase of creating the segment can be completed:

a. Copy /home/rdart/devsrc2/RDA/makeRDASegment to runtime directory.

b. chmod +x makeRDASegment

c. makeRDASegment [runtime path]

5.2.3 RDASRV Segment Build Procedure

After all server changes have been checked in and putback into codemgrtool, the following steps must be performed to complete creation of the segment.

5.2.3.1 Update Segmentation Specific Files

The segment files /home/rdart/devsrc2/RDASRV/SegDescrip/VERSION, /home/rdart/devsrc2/RDASRV/PatchSegDescrip/VERSION and /home/rdart/devsrc2/RDASRV/SegDescrip/ReleaseNotes must be updated with the correct information for the release.

a. From UNIX prompt: su - rdart

b. Enter password.

c. codemgrtool&

d. Highlight work area (e.g., /home/diana/devsrc2/RDASRV).

e. Using right mouse button choose Transactions/Bringover/Update.

f. With right mouse button choose Edit/Select All.

g. With left mouse button click on SegDescrip/ReleaseNotes, SegDescrip/VERSION, and PatchSegDescrip/VERSION.

h. With right mouse button choose Edit/Delete.

i. With left mouse button press Bringover. When complete close status window and Bringover window.

j. From main codemgrtool window double click on work area (e.g., /home/diana/devsrc2/RDASRV) and then double click on SegDescrip.

k. Using both mouse buttons click on ReleaseNotes and VERSION.

l. With right mouse button choose Commands/Check Out.

m. Double click on "..", double click on PatchSegDescrip. Click on VERSION and with right mouse button choose Commands/Check Out as in step 12.

n. Using UNIX editor of choice, edit ReleaseNotes and VERSION files with appropriate changes for the release. This information includes items such as release, version, build date, patch number, and delivery date.

o. From UNIX prompt: codemgrtool&

p. Using left mouse button double click on work area and then double click on SegDescrip.

q. Using both mouse buttons click on ReleaseNotes and VERSION.

r. With right mouse button choose Commands/Check In.

s. Double click on "..", double click on PatchSegDescrip. Click on VERSION and with right mouse button choose Commands/Check Int as in step 18.

t. Using right mouse button choose Transactions/Bringover/Putback.

u. With right mouse button choose Edit/Select All.

v. With left mouse button click on SegDescrip/ReleaseNotes, SegDescrip/VERSION, and PatchSegDescrip/VERSION.

w. With right mouse button choose Edit/Delete.

x. With left mouse button press Putback. When complete close status window and Putback window.

5.2.3.2 Create and Populate Supporting Segmentation Directories and Files

a. Copy /home/rdart/devsrc2/RDASRV/makeRDASRVSegment to runtime directory.

b. chmod +x makeRDASRVSegment

c. makeRDASRVSegment [runtime path]

d. After this step an RDASRV and RDASRV.PATCH directories have been created. The RDASRV.PATCH has to be renamed to reflect the correct patch number for the release (e.g., mv RDASRV.PATCH RDASRV.P11).

5.2.4 Notify Configuration Management

After both segments are built, notify Configuration Management to have the segment tapes built for installing and testing the new runtime.

5.3 Modification Procedures

This paragraph explains how RDA should be modified following receipt of a Global Software Problem Report (GSPR) or internal Problem Report (PR). The following paragraphs specify information necessary to perform modifications.

5.3.1 Supporting Software

The following software packages are required for RDA use:

a. ORACLE RDBMS 7 Server Release 7.1.4 or later w/distributed option,
b. ORACLE PL/SQL Release 2.1.4 or later
c. ORACLE SQL*Plus Release 3.1.3.5.1 or later,
d. SQL*NET ver 2.0.15.0.0 or later,
e. Latest Release of JOPES Core Database (SMDB Segment), and
f. Solaris 2.3/2.4 or later,
g. Sybase Gain Momentum 3.1-41a, and
h. Sun Teamware.

5.3.2 Databases/Data Files

RDA creates many ORACLE objects during the RDADB segment installation. The files to create these objects have the extension *.sql* and are located in /h/RDASRV/sql and /h/RDASRV/install. All *.sql* files should create a log file in the /tmp directory.

5.3.3 Design, Coding and Other Conventions

The following design and coding standards and procedures apply to modification efforts.

5.3.3.1 Coding Standards

RDA was created in accordance with the following:

a. GCCS JOPES Language Independent Coding Standard,
b. GCCS JOPES GEL Coding/Library Structure Standard,
c. GCCS JOPES Stored Packages and Procedures Standard,
d. GCCS JOPES UNIX Script Standard, and
e. GCCS JOPES User Interface Standard for RDA.

These standards appear in JOPES Software Development Standards, and have been informally provided to the Government.

5.3.3.2 Applicable Procedures

The procedures in the following paragraphs should be used to modify each type of source code file found in RDA.

5.3.3.2.1 Gain Momentum Applications

a. After developer is tasked to work on a PR/GSPR that relates to a Gain application, the user will log into the Gain development environment, check out the application(s), make appropriate changes, and test the changes.

b. After testing is complete, application is checked back into Gain and exported to the developer's rtexport directory. Then he sends a mail message to rdart describing the change and the PR/GSPR that required the change.

c. Finally, the PR Tracking system is updated with the changes that were accomplished.

5.3.3.2.2 SQL Scripts

a. After developer is tasked to work on a PR/GSPR that relates to a stored procedure, the user will bring up codemgrtool, bringover the script, check it out, make appropriate changes, compile and test the changes.

b. After testing is complete, the developer checks the script back into codemgrtool and then does a putback. Since an automatic mail message is sent to rdart when the putback is accomplished, no other message is required. However, he has to ensure that the comments written during the putback reflect the PR/GSPR and the changes that were done.

c. Finally, the PR Tracking system is updated with the changes that were accomplished.

5.3.3.2.3 UNIX Scripts

a. After developer is tasked to work on a PR/GSPR that relates to a UNIX script, the user will bring up codemgrtool, bringover the script, check it out, make appropriate changes, and test the changes.

b. After testing is complete, the developer checks the script back into codemgrtool and then does a putback. Since an automatic mail message is sent to rdart when the putback is accomplished, no other message is required. However, he has to ensure that the comments written during the putback reflect the PR/GSPR and the changes that were done. After exiting codemgrtool, he has to change directories to the location within the rdart home area that contains the script and change the permissions on the script to executable.

c. Finally, the PR Tracking system is updated with the changes that were accomplished.

5.3.4 Compilation/Build Procedures

Build the RDA runtime using the process specified in Paragraph 5.2.

5.3.5 Integration and Testing Procedures

To perform integration and test of RDA modification, the RDA System must be properly initiated. On a GCCS client workstation, click on the **{RDA icon}** on the GCCS desktop, or select RDA from the JNAV system. Follow the predetermined test plan and compare the results with those expected.

5.4 Computer Hardware Resources Utilization

Refer to Appendix F, Hardware Utilization, for this information.

6.0 REQUIREMENTS TRACEABILITY

Refer to Appendix C, Requirements Traceability, for this information.

7.0 ACRONYMS

The acronyms used in this document are described below.

AHQ	Ad Hoc Query
ALD	Available to Load Date
ASCII	American Standard Code for Information Interchange
CAS	Crisis Action System
CCC	Cargo Category Code
CIN	Cargo Increment Number
CM	Configuration Management
COA	Course of Action
DART	Dynamic Analysis and Replanning Tool
DATEBT	Requirement Date Transaction
DLFMDT	Force Module Delete ID Transaction
DID	Data Item Description
DISA	Defense Information Systems Agency
EAD	Earliest Arrival Date
EIC	Equipment Identification Code
ESI	External System Interface
FAPES	Force Augmentation Planning and Execution System
FDBM	Functional Database Manager
FIC	Force Indicator Code
FM	Force Module
FRN	Force Requirement Number
FU	Functional Unit
GCCS	Global Command and Control System
GEL	Gain Extension Language
GEO	Geographic
GEOLOC	Geographic Location
GSORTS	GCCS Status of Resources and Training System
GSPR	GCCS Software Problem Report
HP	Hewlett-Packard
ID	Identification
ILOC	Intermediate Location
IRM	Information Resource Management
JDS	Joint Deployment System
JEPES	Joint Engineer Planning and Execution System
JFAST	Joint Flow and Analysis System for Transportation
JJDSDT	Force Module Requirement Transaction
JNAV	JOPES High Level System Navigation
JOPES	Joint Operation Planning and Execution System
JOPESREP	JOPES Reporting Structure
JOPS	Joint Operation Planning System
LAD	Latest Arrival Date
LOGSAFE	Logistics Sustainment Analysis and Feasibility Estimator

LU	Logical Unit
MB	Megabyte
MEPES	Medical Planning and Execution System
MOTIF	Message Oriented Text Interchange Format
NRNUBT	Non-Unit Requirement Transaction
OODB	Object-Oriented Database
OPLAN	Operation Plan
PAX	Passengers
PDR	Pre-Defined Reports
PID	Plan Identification Number
PIN	Personnel Increment Number
PLNUAT	Plan Transaction
POD	Port of Debarkation
POE	Port of Embarkation
PORTS	Port Characteristics
RDA	Requirements Development and Analysis
RDBMS	Relational Database Management System
RDD	Required Delivery Date
RFP	Reference File Paging
RLD	Ready-to-Load Date
S&M	Scheduling and Movement
SDP	Software Development Plan
SPM	Software Programmer's Manual
SQL	Structured Query Language
SUM	Software Users Manual
TCC	Transportation Component Command
TDS	Transaction Distribution System
TP	Transaction Processing
TPFDD	Time-Phased Force and Deployment Data
TUCHA	Type Unit Characteristics
TUDET	Type Unit Equipment Detail
UIC	Unit Identification Code
ULN	Unit Line Number
ULNUBT	Force Requirement Transaction
URP	User Review Panel
USERID	User Identification
UTC	Unit type Code
WIN	WWMCCS Intercomputer Network
WWMCCS	Worldwide Military Command and Control System
XTP	External Transaction Processor